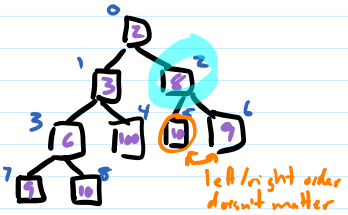
Binary Heaps – binary tree

Shape: all levels filled, except last, all leaves as far left as possible

Order: value in parent ≤ values in children (min-heap)
(priority)
(so lowest priority must be in root



Shape allows storage in an array – better constant factors

curr = 5

parent(curr) = $\lfloor \frac{curr-1}{2} \rfloor$

left(curr) = 2·curr + 1

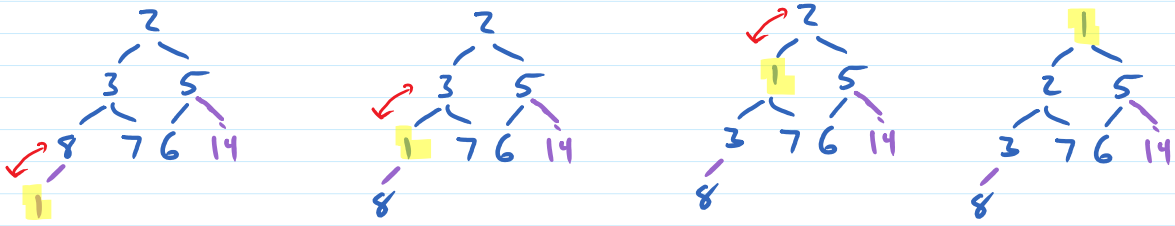
right(curr) = 2·curr + 2

enqueue(key, priority)

add (key, priority) to end
curr = end
while curr ≠ root and curr priority < parent priority
  swap curr, parent
  curr ← parent
] O(1) work per iteration

height of heap

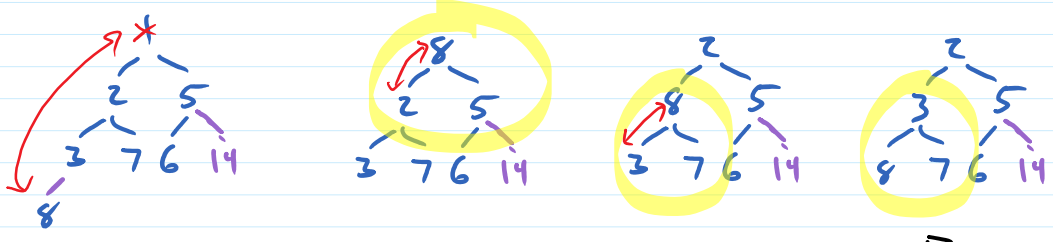
$O(h) = O(\log n)$ iterations



extract-min
dequeue()

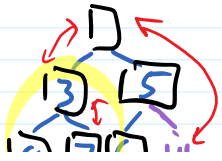
result ← root
move end to root
curr = root
while curr is not a leaf and curr > child
  swap curr, smallest child
  curr ← new location of curr
] O(1)

$O(h) = O(\log n)$ iterations



decrease_priority(item, pri)

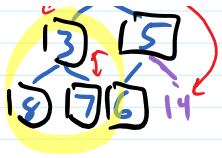
a) look up int index for item (balanced BST)

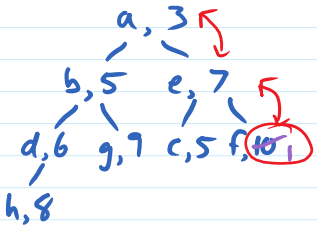


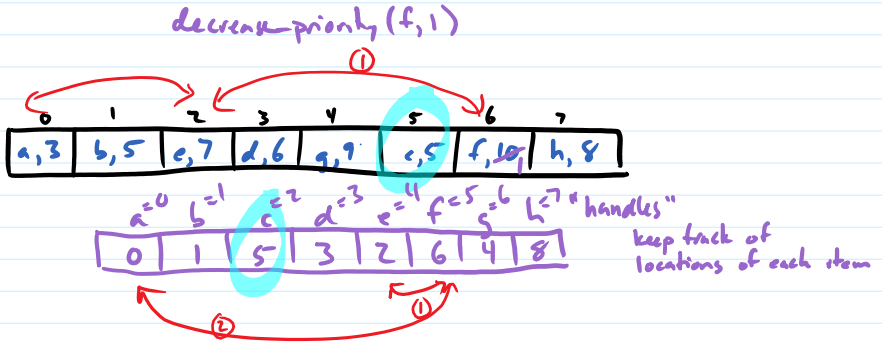
decrease_priority(item, pri)

0) look up int. index for item (balanced BST)

1) find item
2) change priority
3) same loop as in enqueue (starting with curr ← loc of item)

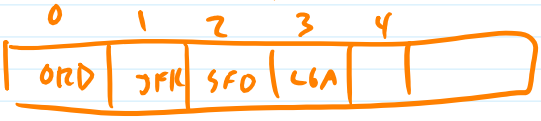






use balanced BST to map items to integer indices
adds O(log n) per operation

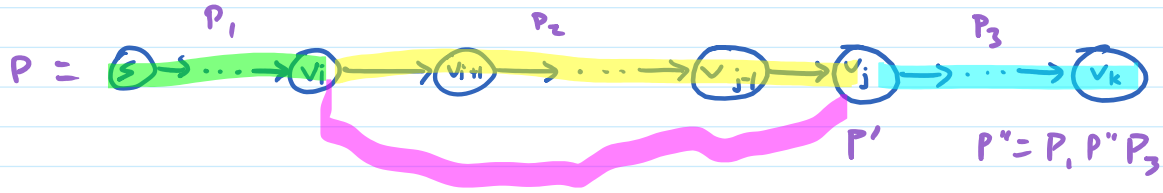
key	index
ORD	0
JFK	1
SFO	2
LGA	3



Optimal Substructure

If $s, v_1, v_2, \ldots, v_k$ is a shortest path $s \rightsquigarrow v_k$ then

for each v_i, v_j $v_i, v_{i+1}, \ldots, v_j$ is shortest path $v_i \rightsquigarrow v_j$



Let P' be any other path $v_i \rightsquigarrow v_j$

$\ell(P_1) + \ell(P_2) + \ell(P_3) = \ell(P) \leq \ell(P'') = \ell(P_1) + \ell(P') + \ell(P_3)$

P is shortest path

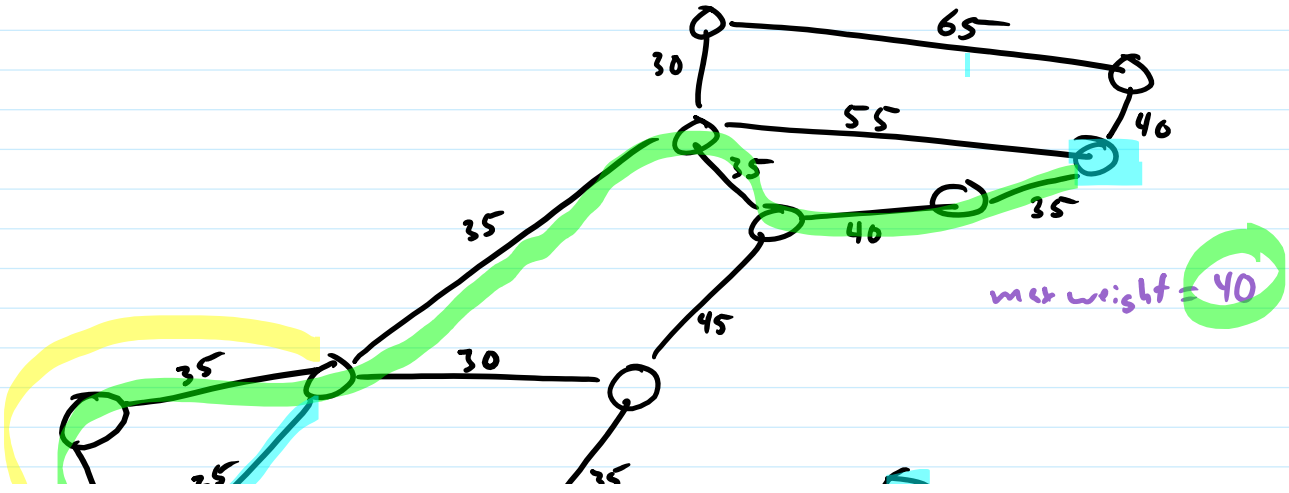
$\ell(P_2) \leq \ell(P')$

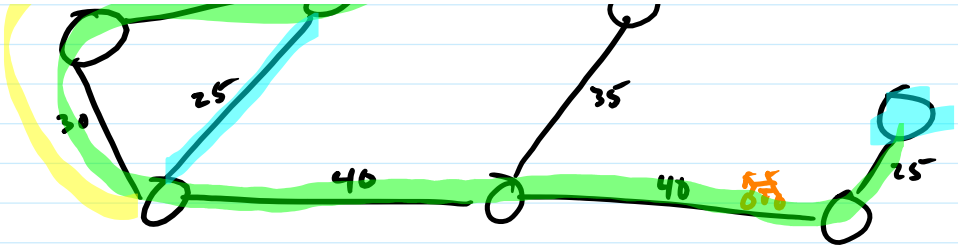
P_2 is shorter than any other path $v_i \rightsquigarrow v_j$

P_2 is shortest path $v_i \rightsquigarrow v_j$

Weaker: If $P = s, v_1, \ldots, v_i, \ldots, v_j, \ldots, v_k$ and P_{ij} is best path $v_i \rightsquigarrow v_j$ then

$s, v_1, \ldots, v_i \overset{P_{ij}}{\rightsquigarrow} v_j, \ldots, v_k$ no worse than P




30
25
40
35
40
25

Invariant

```
Dijkstra (G, ℓ)
    Q ← ∅
    for each vertex v
        d'[v] ← ∞
        π[v] ← NIL
    d'[s] = 0
    for each v
        enqueue(Q, v, d'[v])

    while Q ≠ ∅
        v = extract_min(Q)
        d[v] ← d'[v]
        S ← S ∪ {v}
        for (v,w) ∈ E where w ∈ Q
            if d[v] + ℓ(v,w) < d'(w)
                d'[w] = d[v] + ℓ(v,w)
                π[w] = v
                decrease_priority(Q, w, d'[w])
```

INVARIANT

a) $|S| \geq 1 \rightarrow s \in S$

b) S, Q partition V

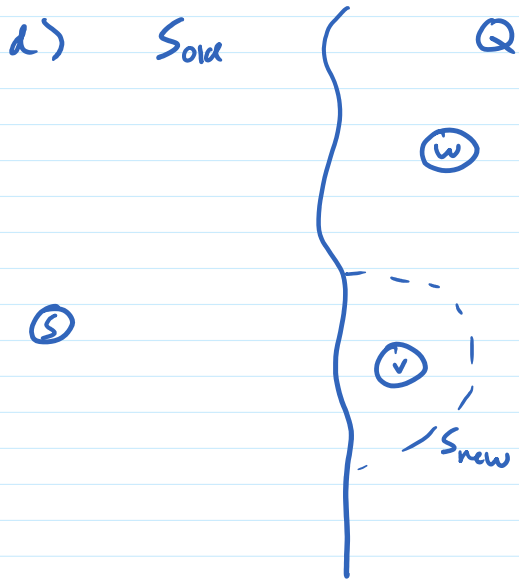
c) $|Q| = n -$ # iterations of loop

d) for $v \in S$, $d[v] = \delta(s,v)$ (total weight of shortest path $s \rightsquigarrow v$)
$\pi[v]$ = next-to-last on that path

e) for $v \in Q$, $d'[v]$ = total weight of shortest path $s \rightsquigarrow v$ using intermediate vertices in S
$\pi[v]$ = next-to-last on that path

f) for all v, $d'[v]$ is the priority of v in Q

g) for all v, $\pi[v] =$ NIL or $\pi[v] \in S$



c) $d[v]$ is total weight of a shortest path $s \rightsquigarrow v$ (for all $v \in S$)
(min tot weight)